Package ‘leiden’

February 4, 2020

Type Package

Title R Implementation of Leiden Clustering Algorithm

Version 0.3.3

Date 2020-02-03

Description Implements the 'Python leidenalg' module to be called in R.
Enables clustering using the leiden algorithm for partition a graph into communities.
See the 'Python' repository for more details: <https://github.com/vtraag/leidenalg>
Traag et al (2018) From Louvain to Leiden: guaranteeing well-
connected communities. <arXiv:1810.08473>.

License GPL-3 | file LICENSE

URL `https://github.com/TomKellyGenetics/leiden`

Imports methods, reticulate, Matrix, igraph

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Suggests data.table, tibble, devtools, covr, testthat, spelling,
knitr, rmarkdown, RColorBrewer

Language en-US

VignetteBuilder knitr

NeedsCompilation no

Author S. Thomas Kelly [aut, cre, trl],
Vincent A. Traag [com]

Maintainer S. Thomas Kelly `<tom.kelly@riken.jp>`

Repository CRAN

Date/Publication 2020-02-04 07:40:06 UTC

R topics documented:

leiden *Run Leiden clustering algorithm*

Description

Implements the Leiden clustering algorithm in R using reticulate to run the Python version. Requires the python "leidenalg" and "igraph" modules to be installed. Returns a vector of partition indices.

Usage

```
leiden(
  object,
 partition_type = c("RBConfigurationVertexPartition", "ModularityVertexPartition",
    "RBERVertexPartition", "CPMVertexPartition", "MutableVertexPartition",
    "SignificanceVertexPartition", "SurpriseVertexPartition"),
  initial_membership = NULL,
  weights = NULL,
  node_sizes = NULL,
  resolution_parameter = 1,
  seed = NULL,
  n_iterations = 2L
)
```

Arguments

`object`	An adjacency matrix compatible with `igraph` object or an input graph as an `igraph` object (e.g., shared nearest neighbours).
`partition_type`	Type of partition to use. Defaults to RBConfigurationVertexPartition. Options include: ModularityVertexPartition, RBERVertexPartition, CPMVertexPartition, MutableVertexPartition, SignificanceVertexPartition, SurpriseVertexPartition (see the Leiden python module documentation for more details)
`initial_membership, weights, node_sizes`	Parameters to pass to the Python leidenalg function (defaults initial_membership=None, weights=None). Weights are derived from weighted igraph objects and non-zero integer values of adjacency matrices.
`resolution_parameter`	A parameter controlling the coarseness of the clusters
`seed`	Seed for the random number generator. By default uses a random seed if nothing is specified.
`n_iterations`	Number of iterations to run the Leiden algorithm. By default, 2 iterations are run. If the number of iterations is negative, the Leiden algorithm is run until an iteration in which there was no improvement.

Value

A partition of clusters as a vector of integers

Examples

```
#check if python is availble
modules <- reticulate::py_module_available("leidenalg") && reticulate::py_module_available("igraph")
if(modules){
#generate example data
adjacency_matrix <- rbind(cbind(matrix(round(rbinom(4000, 1, 0.8)), 20, 20),
                                matrix(round(rbinom(4000, 1, 0.3)), 20, 20),
                                matrix(round(rbinom(400, 1, 0.1)), 20, 20)),
                          cbind(matrix(round(rbinom(400, 1, 0.3)), 20, 20),
                                matrix(round(rbinom(400, 1, 0.8)), 20, 20),
                                matrix(round(rbinom(4000, 1, 0.2)), 20, 20)),
                          cbind(matrix(round(rbinom(400, 1, 0.3)), 20, 20),
                                matrix(round(rbinom(4000, 1, 0.1)), 20, 20),
                                matrix(round(rbinom(4000, 1, 0.9)), 20, 20)))
rownames(adjacency_matrix) <- 1:60
colnames(adjacency_matrix) <- 1:60
#generate partitions
partition <- leiden(adjacency_matrix)
table(partition)

#generate partitions at a lower resolution
partition <- leiden(adjacency_matrix, resolution_parameter = 0.5)
table(partition)

#generate example weights
weights <- sample(1:10, sum(adjacency_matrix!=0), replace=TRUE)
partition <- leiden(adjacency_matrix, weights = weights)
table(partition)

#generate example weighted matrix
adjacency_matrix[adjacency_matrix == 1] <- weights
partition <- leiden(adjacency_matrix)
table(partition)


# generate (unweighted) igraph object in R
library("igraph")
adjacency_matrix[adjacency_matrix > 1] <- 1
snn_graph <- graph_from_adjacency_matrix(adjacency_matrix)
partition <- leiden(snn_graph)
table(partition)

# generate (weighted) igraph object in R
library("igraph")
adjacency_matrix[adjacency_matrix >= 1] <- weights
snn_graph <- graph_from_adjacency_matrix(adjacency_matrix, weighted = TRUE)
partition <- leiden(snn_graph)
table(partition)

# pass weights to python leidenalg
adjacency_matrix[adjacency_matrix >= 1 ] <- 1
snn_graph <- graph_from_adjacency_matrix(adjacency_matrix, weighted = NULL)
weights <- sample(1:10, sum(adjacency_matrix!=0), replace=TRUE)
partition <- leiden(snn_graph, weights = weights)
table(partition)

# run only if python is available (for testing)
}
```

Index